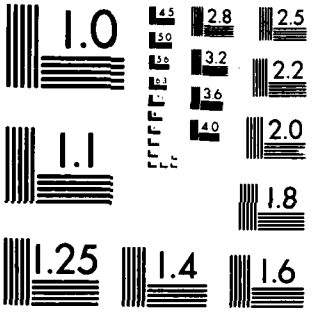
AD-A135 596 FAST COLOUR DISPLAYS OF DIGITAL TERRAIN ELEVATION DATA (U) ELECTRONICS RESEARCH LAB ADELAIDE (AUSTRALIA) M J DOWLING ET AL. NOV 86 ERL-0393-TM 1/1

UNCLASSIFIED F G 12/5 NL

END
DATE
FILMED
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(12)

DTIC FILE COPY

ERL-0393-TM

AR-004-868

AD-A185 596

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

ELECTRONICS RESEARCH LABORATORY

DEFENCE RESEARCH CENTRE SALISBURY
SOUTH AUSTRALIA

TECHNICAL MEMORANDUM

ERL-0393-TM

FAST, COLOUR DISPLAYS OF DIGITAL TERRAIN ELEVATION DATA

M.J. DOWLING and S.C.C. McAULEY

Original contains color plates: All DTIC reproductions will be in black and white

DTIC ELECTE OCT 0 6 1987 S D

Technical Memoranda are of a tentative nature, representing the views of the author(s), and do not necessarily carry the authority of the Laboratory.

DISTRIBUTION STATEMENT A
Approved for public release
Distribution Unlimited

Approved for Public Release

COPY No. 28

© Commonwealth of Australia

NOVEMBER 1986

AR-004-868

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

ELECTRONICS RESEARCH LABORATORY

TECHNICAL MEMORANDUM

ERL-0393-TM

FAST, COLOUR DISPLAYS OF DIGITAL TERRAIN ELEVATION DATA

M.J. Dowling and S.C.C. McAuley

S U M M A R Y

Software has been developed to provide the following displays of digital terrain elevation data of a given map area:

(a) A contour plot with one or more visibility diagrams,

(b) An oblique view of the map area from any azimuth or elevation angle,

(c) A perspective view of a specified sector from a selected viewpoint, and

(d) A point-to-point line-of-sight profile.

Emphasis has been placed on making the software easy to use, on generating the displays as quickly as possible, and on making effective use of colour.

POSTAL ADDRESS: Director, Electronics Research Laboratory,
Box 2151, GPO, Adelaide, South Australia, 5001.

TABLE OF CONTENTS

		Page
1.	INTRODUCTION	1
2.	OVERVIEW OF THE SOFTWARE	1
3.	DETAILED DESCRIPTION OF EACH DISPLAY	3
	3.1 Contour plot (with optional visibility diagrams)	3
	3.1.1 Erase the latest visibility diagram	4
	3.1.2 Draw a visibility diagram	4
	3.1.3 Draw more contour lines on the same display	4
	3.1.4 Change the scale of the contour display	5
	3.1.5 Move to a different area of the same map	5
	3.1.6 Add or remove a grid	5
	3.1.7 Erase the screen and stop, or simply stop	5
	3.2 Oblique view	5
	3.3 Perspective view	6
	3.3.1 Range increments	8
	3.3.2 Vertical field-of-view	9
	3.3.3 Vertical position on the screen	10
	3.3.4 Visibility	10
	3.4 Profile	11
4.	THE ALGORITHMS	12
	4.1 Contour plotting	12
	4.2 Visibility diagram	14
	4.3 Oblique view	17
	4.3.1 Scale	18
	4.3.2 Transform	20
	4.3.3 Mask	21
5.	DETAILED DESCRIPTION OF THE USER AND GRAPHICS INTERFACES	22
	5.1 Menus	23
	5.1.1 Functional description	23
	5.1.2 Implementation	23
	5.2 User terminal input	23
	5.2.1 Functional description	24
	5.2.2 Implementation	24
	5.3 Graphics line driver	25
	5.3.1 Functional description	25
	5.3.2 Implementation	26

Page

5.4 Software interrupt 27

5.4.1 Functional description 27
5.4.2 Implementation 28

5.5 Program termination 28

6. POSSIBLE ENHANCEMENTS 29

6.1 Line diagrams 29

6.2 Colour options 29

6.2.1 Oblique and perspective view 29
6.2.2 Contour plot 29

6.3 Additional information 30

6.4 Overlap of adjoining areas 30

6.5 Panning (of contour plot or perspective view) 31

7. CONCLUSION 31

REFERENCES 33

LIST OF APPENDICES

I CRITIQUE OF THE ASR SOFTWARE 35

II MODULE DEPENDENCIES 37

III FILE CONTENTS 39

LIST OF FIGURES

1. Contour Plot: Normal map scale, 20 m contours drawn using every point of the terrain data (1 min 50 s)

2. Contour Plot: Half map scale, 20 m contours drawn using every point of the terrain data (7 min 40 s)

3. Contour Plot: Half map scale, 50 m contours drawn using every point of the terrain data (3 min 10 s)

4. Contour Plot: Half map scale, 50 m contours drawn using every fifth point of the terrain data (30 s)

5. Contour Plot: Half map scale, 20 m contours drawn using every fifth point of the terrain data (66 s). Also, a visibility diagram drawn with a 1° increment (18 s), and a grid (1 s)

6. Oblique view from the South at 30° elevation, using every second point of the terrain data (1 min 40 s)

7. Oblique view from the South at 30° elevation, using every fifth point of the terrain data (20 s)

8. Oblique view from the Northwest at 15° elevation, using every fifth point of the terrain data (20 s)

9. Oblique view from the Northwest at 15° elevation, using every fifth point of the terrain data, with no vertical exaggeration (20 s)

10. Perspective view (35 s)

11. Profile (3 s)

12. Perspective view (constant range increments)

13. ASR Contour Plotting Algorithm

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist Special
A-1

1. INTRODUCTION

This memorandum describes a software package which has been developed for multi-colour displays of digital terrain elevation data. Emphasis has been placed on producing a package which is easy to use and which uses efficient algorithms, thereby enabling the displays to be generated as quickly as possible. The work was initiated as a result of a request by Army Office to investigate techniques available for the storage and display of map information for tactical military use.

The original intention for satisfying the Army requirement was to modify or adapt existing software packages and, to this end, Army Survey Regiment(ASR) Bendigo made available the package described in reference 1. An assessment was carried out of the package and it was found to have a significant number of limitations which are outlined in Appendix I. Rather than attempt the major task of modifying this package it was decided to develop a completely new package which is the subject of this report. In view of the experience already gained by the authors this was considered to be a more cost effective approach than evaluating further packages.

The basic input data for the displays is Digital Terrain Elevation (DTE) data which is currently being generated for large areas of Australia by ASR, and consists of spot heights of the terrain recorded at the intersection of a uniform two-dimensional grid superimposed on the area to be processed. The software package uses this data to produce the following displays:

(a) A contour plot with one or more visibility diagrams (figures 1 to 5),

(b) An oblique view of the whole map area (figures 6 to 9),

(c) A perspective view of a specified sector from a selected viewpoint (figure 10), and

(d) A point-to-point line-of-sight (LOS) profile (figure 11).

The software has been developed using Fortran 77 on a PDP-11/34 with a Vectrix colour graphics screen. All Vectrix commands are contained within a small set of graphics subroutines, which should enable easy modification for a different colour graphics screen.

Appendix II contains tables of the module dependencies for each of the four main programs, and Appendix III lists the file names and the contents of each file.

2. OVERVIEW OF THE SOFTWARE

The fundamental approach has been to identify simple algorithms which enable each display to be drawn as quickly as possible. In addition, for the first three programs (above) the user can choose to use:

- Every point of the DTE data to draw the display, which gives maximum detail but takes much longer, or

- Every nth point of the DTE data, where n is a number greater than 1. For example, if n was given the value 5, then the display would be drawn using every 5th point of the DTE data (eg at 500 m intervals instead of 100 m), giving a reasonable representation much more quickly. Compare figure 5 with figure 2, with the time taken to draw each display shown within brackets in the title of the figure.

Thus preliminary displays can be drawn very quickly using every 5th point of the terrain data, and when required, a particular display can be re-drawn in more detail.

Emphasis has been placed on making the package easy-to-use. Initially the user is presented with a menu of the available terrain data bases. When a particular area is selected, another menu will appear which enables the user to:

- Select one of the four types of display, or
- Return to the primary menu.

When a particular type of display is selected, simple instructions prompt the user to supply the necessary information. If the user input contains invalid characters or is out of range, the user is informed and asked to re-enter the data. Because the user may realise partway through that the display currently being generated is not what he requires, an interrupt key (Ctrl C) has been provided, to enable an immediate return to the secondary menu. Each display has headings which record all the information required to reproduce the display later.

The terrain data base required is standard Digital Terrain Elevation (DTE) data, provided by the Army Survey Regiment (Bendigo). The Army Survey Regiment can provide the data at a range of specified data spacings; a small data spacing means that much more data must be stored to cover a given area, but more detailed displays are possible. However,

(a) for practical use in the field, fast approximate displays may be of greater value than detailed displays which take a long time to generate, and

(b) the level of detail may in practice be limited by the precision of the screen.

A data spacing of 100 m is recommended as a reasonable compromise; for a typical 1:50 000 scale map, covering an area of 27 km x 24 km the storage requirement is 64 800 elevation values, which is acceptable. The spacing of points on the screen is 2 mm at a scale of 1:50 000 or 4 mm at a scale of 1:25 000. Within a square of that size, linear interpolation (equivalent to assuming a constant slope from corner to corner) is perfectly adequate.

There are a number of significant features of the contour plot display, namely:

(a) the contour lines are drawn in colour, with a simple coding system which enables the height at a particular point to be easily and accurately determined,

(b) the contour plot can be drawn at normal map scale, half map scale, or twice map scale,

(c) a grid (1 km at a normal map scale of 1:50 000) can be overlaid onto the contour plot to assist in reading the coordinates of a point, and, if no longer required, can be removed without affecting the contour lines,

(d) a visibility diagram can be drawn, removed (without affecting the contour lines), or remain on the screen while another visibility diagram is drawn, etc.

3. DETAILED DESCRIPTION OF EACH DISPLAY

The user must select the required terrain data from the primary menu, and then select one of the following displays from the secondary menu.

3.1 Contour plot (with optional visibility diagrams)

The user is asked to specify the following:

(a) The scale at which the display is to be drawn on the screen (normal map scale, half map scale, or twice map scale); the default is normal map scale.

(b) The 8-digit coordinates of the south-west corner of the "viewing window" (the section of the current map which will be displayed on the screen); the default coordinates are the south-west corner of the terrain data.

At this stage the program draws the map name at the top of the screen, the boundary of the viewing window on the screen, and the coordinates of the four corners.

The user must then specify:

(c) The interval (in metres) between displayed contours; the default value is 20 m.

(d) The terrain data sampling factor, N, which tells the program to use every Nth point of the terrain data to draw contour lines; the default value of N is 5.

The program then proceeds to draw the contour lines , using the algorithm described in Section 5. The effect of varying (c) and (d) is shown in figures 1 to 5; in particular, compare figures 3 and 4 - the reduced level of detail in figure 4 cuts the drawing time from 3 min 10 s to 30 s.

The default colour for contour lines is blue; contours which are a multiple of 100 m are colour-coded as follows:

cyan	0 m
orange	100 m
green	200 m
red	300 m
yellow	400 m

The colours are repeated for 500 m to 900 m respectively, and so on. A wider range of colours could be used; only a limited number of colours could easily be resolved on the system used to develop the software. If colour hard-copy is available, then a check should be made of whether the hard-copy colours are different to those on the screen - if so, the colours chosen need to be easily distinguished both on the screen and on the hard-copy.

When all the contour lines have been drawn, the user is presented with the following list of options:

3.1.1 Erase the latest visibility diagram

If there is more than one visibility diagram on the screen, only the last one drawn can be erased. Similarly, the information on the left hand side of the screen (viewpoint location, height above ground, sector, and maximum range) applies only to the last diagram drawn.

3.1.2 Draw a visibility diagram

In the present version of the program, the user is asked to enter the viewpoint location as 8-digit coordinates, because the Vectrix screen has no local cross-hair control. The program has the necessary coding (commented out) to enable the user to use cross-hair controls on the display device to specify the system location; subroutine LOCATG would need to be created to enable cross-hair input, and read the appropriate position of the cross-hairs.

As well as the location of the viewpoint, the user will also be asked to:

(a) Specify the height above ground of the viewpoint (m); the default value is 10 m.

(b) Specify the maximum range of the visibility diagram (m): for a full circle visibility diagram, the maximum range possible is 5000 m, because of the size of the array used to temporarily store the terrain data needed to draw the visibility diagram. The default value for maximum range is 3000 m.

(c) Select either an arc for the visibility diagram or a full circle, the default is a full circle.

(d) Specify the following inputs if an arc is selected:

- the initial azimuth angle (in degrees clockwise from North), and
- the sweep angle (in degrees).

(e) Finally specify the azimuth angle increment (in degrees) ie the angle between each radial line; the default value is 3°.

3.1.3 Draw more contour lines on the same display

If the initial contour interval was too large, then the gaps can be filled in by selecting a smaller interval. The sampling factor, N, would not normally be changed at this stage.

There may be a need to simply draw one or two extra contour levels; the present version of the program does not allow this, but it could be incorporated by adding another option, and asking the user to enter:

- an initial contour level (h_1),
- a final contour level (h_2), and
- an increment.

Thus one extra contour level could be drawn (if $h_1 = h_2$) or a limited number of extra contour levels.

3.1.4 Change the scale of the contour display

The procedure in Section 3.1 is repeated. When the user has entered a new scale and new coordinates for the south-west corner of the viewing window, then the screen is erased, and the title, border and new corner coordinates are drawn.

3.1.5 Move to a different area of the same map

The user enters new coordinates for the south-west corner of the viewing window, then the screen is erased and the title, border, and new corner coordinates are drawn. Then the procedure in Section 3.1 is repeated from step (c).

3.1.6 Add or remove a grid

A grid with a spacing equivalent to 2 cm on the screen is superimposed on the viewing window. Thus at a scale of 1:50 000 the grid spacing is 1 km, at 1:25 000 the grid spacing is 500 m, and at 1:100 000 the grid spacing is 2 km. Replace-complement mode is used, so that the colour is the complement of what was previously on the screen. The original display can be completely restored, simply by drawing the grid again.

3.1.7 Erase the screen and stop, or simply stop

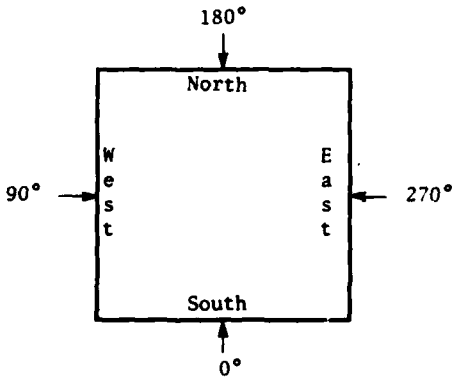
This returns the user to the secondary menu.

3.2 Oblique view

The user is asked to specify the following:

(a) The sampling factor, N, which tells the program to use every Nth point of the terrain data to draw the display; the default value is 5. Note that a sampling factor of 1 or 2 will probably exceed the resolution of the screen, giving a very dense display (which takes a long time to draw).

(b) The azimuth viewing angle (in degrees clockwise from North); as shown below, an azimuth viewing angle of 0° generates a view of the terrain data from the South side. The default value is 0°.



Azimuth Viewing Angle

(c) The elevation viewing angle (in degrees) in the range from 0 to 90°; the default value is 30°.

(d) The vertical exaggeration: variations in terrain height are usually small relative to horizontal distances. A vertical exaggeration factor is used to make the variations in terrain height more easily visible. The default value is 3, which is a commonly used value; if the terrain is largely flat or more detail is required, then a bigger value may be better. Figure 8 shows a view of Christmas Island with the default vertical exaggeration; figure 9 shows the same view with no vertical exaggeration.

The program then draws a boundary around the screen, the map name and the values specified (above) and proceeds to generate the required display. If the user realises that this display is not what he really wants, Ctrl C will interrupt the program and proceed to the next prompt (which normally appears when the display is complete). Here the user has two options: select another view, or return to the secondary menu.

Each oblique view is automatically scaled, such that either

- the maximum width equals the width of the screen, or
- the maximum height equals the height of the screen.

If the user decides to select another view, then when all of the information required (above) has been entered, the screen is erased, the boundary, map name and view information is drawn, and then the new oblique view.

3.3 Perspective view

The user is asked to specify the following:

(a) The 8-digit coordinates of the viewing position; the default value is the south-west corner of the terrain data.

(b) The minimum range to be displayed on the screen: the program will check for visibility all the way from the viewing position out to the maximum range, but will only start to display the view from the nominated minimum range onwards. The default value is 100 m.

(c) The maximum range (m); the default value is 5000 m. The limit for the maximum range depends upon the capacity of the computer and the field-of-view, as explained following (e) below.

(d) The azimuth angle at the centre of the required view (in degrees clockwise from North); the default value is 0°.

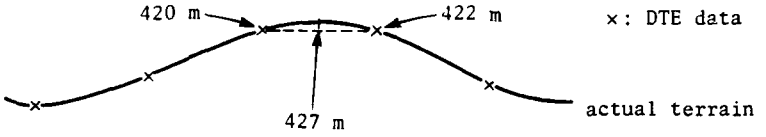
(e) The field-of-view (in degrees); the default value is 90°.

The program identifies the smallest rectangular subset of terrain data which encloses the specified sector, and reads that data into a two-dimensional array: if the array is not large enough, a message will be displayed telling the user to:

- decrease the maximum range, or
- increase the value of NZ in subroutines MAPDAT and PVIEW.

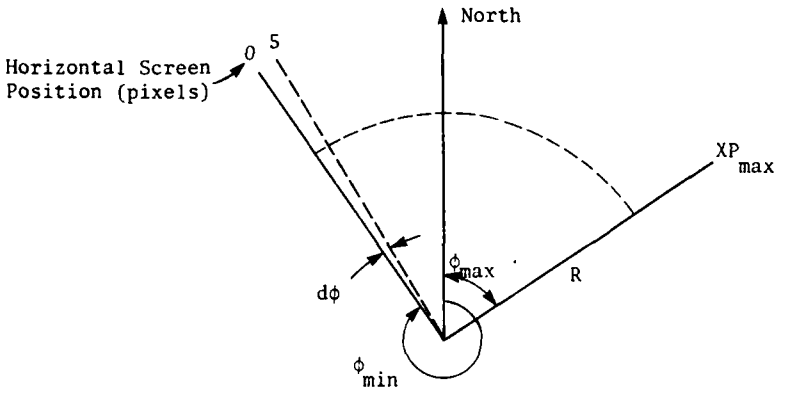
The program then calculates and displays the terrain height at the viewpoint, and asks the user to enter a height above ground (m); the default value is 10 m.

Note that linear interpolation of terrain height data may have the effect of flattening the tops of hills. For example, if the viewpoint is on top of a hill where the exact height is shown on the map (eg 427 m), the program may, as shown below, calculate the height at the viewpoint as 421 m. In this case, add 6 m to the normal height above ground, to compensate for the flattening which has been introduced by the use of DTE data.



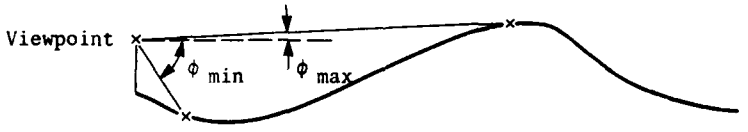
The perspective view is produced by mapping:

- azimuth angle into horizontal position on the screen (pixels)



(the azimuth increment dϕ is calculated to correspond to 5 pixels on the screen), and

- elevation angle into vertical position on the screen



(there are several options for setting the vertical scaling of the view; these will be explained later in this Section).

3.3.1 Range increments

The program forms the perspective view display by drawing a number of lines across the screen, at range values varying from minimum range to maximum range. At present, the average spacing of the lines (ΔR) is set equal to the spacing of the DTE data (GRDSPC). Thus the number of lines is given by:

$$N_R = \frac{R_{max} - R_{min}}{\Delta R}$$

Note that an increase in the allowable maximum range may result in an excessive number of lines; if such is the case then the following scheme is suggested:

$$\text{if} \quad \frac{R_{max} - R_{min}}{GRDSPC} \leq 50 \text{ , set } \Delta R = GRDSPC,$$

$$\text{if} \quad 50 < \frac{R_{max} - R_{min}}{GRDSPC} \leq 100 \text{ , set } \Delta R = 2*GRDSPC,$$

etc.

Thus there will never be more than 50 lines across the screen.

If $R_{max} - R_{min}$ is not a multiple of GRDSPC, then so that a line is drawn at the maximum range, ΔR is adjusted, as follows: firstly,

$$N_R = \left[\frac{R_{max} - R_{min}}{\Delta R} \right]$$

where the square brackets denote integer truncation, then ΔR is reset:

$$\Delta R = \frac{R_{max} - R_{min}}{N_R}$$

It was found that constant range increments led to lines in the distance being too close together, sometimes exceeding the resolution of the display device (figure 12). As a result, the program now uses a range increment which varies linearly from $0.5\Delta R$ at minimum range (R_{min}) to $2\Delta R$ at maximum range (R_{max}) (figure 10).

3.3.2 Vertical field-of-view

The program makes two passes through each range and each angle within the field-of-view. The first pass is used to determine the vertical range of the terrain within the field-of-view (θ_{min} to θ_{max}). This pass does not check for visibility.

The user then has two options, namely,

(a) To adjust the scale so that the view fills the screen vertically: θ_{min} and θ_{max} are left unchanged, or

(b) To set the vertical field-of-view (FOV) to:

$$\frac{\text{horizontal FOV}}{V}$$

where V is a vertical exaggeration factor, in the range from 1 to 20. An upper limit has been imposed simply so that the program can check that a valid value has been entered: the value of 20 is an arbitrary choice. In this case, θ_{min} and θ_{max} are reset as follows:

Let

$$\bar{\theta} = (\theta_{min} + \theta_{max})/2$$

and

$$\phi_R = \phi_{max} - \phi_{min} = \text{azimuth range.}$$

If the height of the screen was equal to the width of the screen, then the elevation range, θ_R, would be given (as above) by:

$$\theta_R = \frac{\phi_R}{V}$$

However, normally the height of the screen, H_s, is less than the width of the screen, W_s, so the elevation range which can be displayed is restricted to:

$$\theta_R = \frac{\phi_R * H_s}{V * W_s}$$

If $\bar{\theta}$ is kept at the centre of the screen, then

$$\theta_{min} = \bar{\theta} - \theta_R/2$$

and

$$\theta_{max} = \theta_{min} + \theta_R$$

3.3.3 Vertical position on the screen

Now the vertical position on the screen can be calculated: let R be the horizontal range from the viewpoint (at height $Z_S + H$) to a point with height Z. The angle θ is given by:

$$\theta = TAN^{-1} \frac{Z - (Z_S + H)}{R}$$

and hence, YP the vertical position on the screen (pixels) is given by:

$$YP = NINT ((\theta - \theta_{min})*d\theta)$$

where

$$d\theta = \frac{\text{vertical range of screen (pixels)}}{\theta_{max} - \theta_{min}}$$

3.3.4 Visibility

On the second pass, a mask array, with an entry for every fifth pixel across the screen, is used to determine whether points are visible as follows:

(a) Initially, the mask array is set to zero,

(b) For each range from $0.5\Delta R$, the program checks the height of the point at each successive azimuth angle against the mask; if the current point is visible, then

- the mask at that point is updated, and
- if the range exceeds R_{min}, and the previous point was visible, then a line is drawn between the two points.

A "zoom" effect can be obtained by, for example, setting range limits from 4 to 6 km; note that in this case the program checks for

masking from the minimum range (0.5ΔR) right out to 6 km, but only displays visible lines within the prescribed range limits. On such a display, the accumulated mask at R_{min}, if it obscures any of the subsequent view, will be displayed as a yellow line.

3.4 Profile

The user is asked to specify the following:

(a) The 8-digit coordinates of the start point and the end point, and the height above ground at each of these points; default start and end points are calculated to give a 10 km profile across the middle of the map (the default height above ground is zero at each point).

The program then:

- checks whether the selected profile crosses rows or columns more often,
- steps along the profile, calculating the height at each row/column intersection point,
- determines the minimum and maximum height along the length of the profile, and
- calculates and prints the length of the profile (m).

The user is then asked to specify:

(b) The scale for the horizontal axis (range in metres); the default is for the program to adjust the scale so that the selected profile is 20 cm long on the screen. Alternatively, the user can select from three fixed scales (map scale, half map scale, and twice map scale), except that the program will not allow the user to select a fixed scale for which the length exceeds 20 cm on the screen.

(c) The minimum and maximum values for the vertical axis; the default values are the nearest multiple of 100 m below and above the actual minimum and maximum height (derived from the points along the profile).

(d) The vertical exaggeration; the default value is 3.

If the selected vertical exaggeration causes the vertical axis to be too long for the available space on the screen, the message "Vertical range too large for the screen" is displayed, and the user is prompted to change the limits of the vertical axis, or the vertical exaggeration.

The program draws the title, the axes, and, in green, the profile between the selected start and end points. Also, in red a straight line from the nominated height above ground at the start point to the nominated height above ground at the end point.

The user now has three options, namely:

(a) Change the presentation of the same points; the user can change the height above-ground at the start or the end point, the horizontal scale, the vertical axis, or the vertical exaggeration, without the delay associated with calculating the height at points along the profile.

(b) Erase the screen and draw another profile (ie repeat from step (a) at the beginning of this Section), or

(c) Return to the secondary menu.

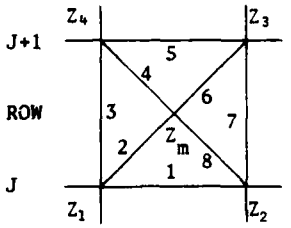
4. THE ALGORITHMS

This Section contains a detailed description of the main algorithms used by the terrain display package.

4.1 Contour plotting

The method used is an adaptation of the method described in reference 3, and can be summarised briefly as follows:

(a) Consider the square shown:



(b) The height at the centre point, Z_m, is approximated by

$$Z_m = (Z_1 + Z_2 + Z_3 + Z_4)/4$$

(c) The line segments connecting the four corner points and the centre point are numbered 1 to 8 as shown.

(d) Consider a contour height, H. Each line segment is tested to see whether the contour intersects the line segment: eg, for segment 1.

Let

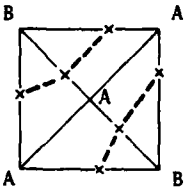
$$T_1 = Z1 - H$$

and

$$T_2 = Z2 - H.$$

If $\Delta = T_1 \times T_2$ is positive, then the entire line segment is either above or below the contour height H. If Δ is negative, then the contour intersects the line segment, and the point of intersection can be found by linear interpolation. As described below, for this application the contour height H is adjusted so that Δ cannot be equal to zero.

(e) When all segments have been checked, the program reorders the points (if necessary) as described in reference 3, and draws the segment(s) of the contour which lie within the square, eg,



A = above contour height
B = below contour height

Note that the algorithm will correctly draw the contour lines in the case where two contours at height H lie within a data square (as shown above).

In order to avoid handling special cases which arise if the contour height is equal to the height at a corner of the data square (or at the middle point), the following procedure was adopted:

- Assuming that the terrain data (ie Z_1, Z_2, Z_3, Z_4) will always be integer values, then
- The height at the centre of the data square, Z_m, will always be a multiple of 0.25.
- Assuming also requested contour height, H_R, will always be an integer value, then the contour height H used within the data square is given by:

$$H = REAL(H_R) - 0.1 \quad \text{if } H_R \neq 0,$$

$$= REAL(H_R) + 0.1 \quad \text{if } H_R = 0.$$

The adjustment of 0.1 m leads to a negligible shift in the position of the contour line, but greatly simplifies the processing within each data square. If the adjustment is negative for $H_R > 0$ (eg H_R=20, H=19.9), then the contour line is drawn around the edge of a hill, for example

```
          17
     18 19 20 16
18   20 20 20 16
     18 20 20 16
        17 16
```

but not around a basin, for example

```
          22
     21 21 20 22
21   20 20 20 22
     21 20 20 22
        21 22
```

and of course the reverse would be true if the adjustment was positive. Assuming that it is more important for hills to be drawn, the negative adjustment has been used for $H_R > 0$.

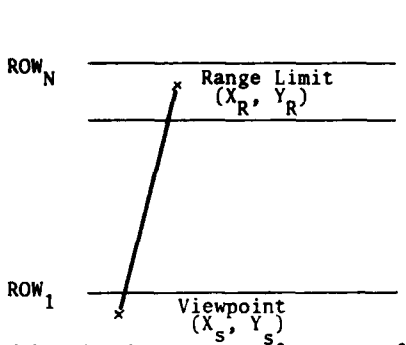
4.2 Visibility diagram

When the user elects to draw a visibility diagram, the program prompts the user for the viewpoint location, sector, and maximum range, and then performs the following activites:

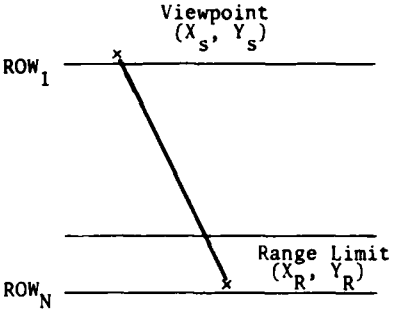
(a) Identifies the smallest rectangular subset of terrain data which encloses the designated area, and reads that data into a two-dimensional array. If the array is not large enough, a message will be displayed telling the user to reduce the range, or recompile the program with a bigger array.

(b) Calculates the terrain height at the viewpoint, from the four surrounding data points, using bilinear interpolation, and sets the plot position at the viewpoint.

The visibility diagram consists of a number of radial lines, drawn at the nominated azimuth angle increment. For each radial line, the program determines whether rows or columns are crossed more often. Note that row and column numbers increment from 1 at the SW corner of the map, and that within the program coordinates such as the viewpoint location (X_s, Y_s) are in metres relative to the SW corner of the map. If rows are crossed more often, (as in the examples shown below) then the program carries out the following procedure:



(a) Azimuth angle ≤ 45° or ≥ 315°



(b) 135° ≤ Azimuth angle ≤ 225°

(a) ROW_1 is set equal to that row between the viewpoint and the range limit which is closest to the viewpoint, and ROW_N to the first row beyond the range limit (except if the range limit is on row R, then $ROW_N = R$).

(b) The following values are calculated for use in determining whether LOS exists to each row intersection point: for ROW_1, as shown below,

$DY_1 = REAL(ROW_1 - 1) * GRDSPC - Y_S$

$DX_1 = DY_1 * TANAZ$

$DR_1 = SQRT(DX_1^2 + DY_1^2)$

and for intermediate rows,

$DY_i = GRDSPC * RSTEP$

$DX_i = DY_i * TANAZ$

$DR_i = SQRT(DX_i^2 + DY_i^2)$

where

(X_s, Y_s) is the viewpoint location,

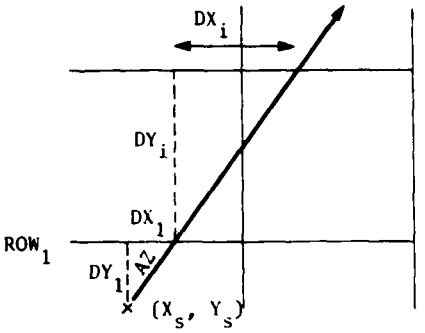
(X_R, Y_R) is the range limit,

GRDSPC is the DTE spacing (m),

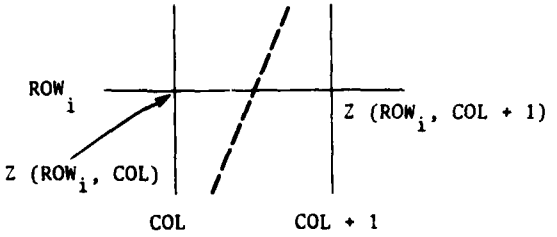
RSTEP = 1. if AZ ≤ 45° or AZ≥ 315°,

= -1. if 135° ≤ AZ ≤ 225°,

TANAZ = TAN(AZ).



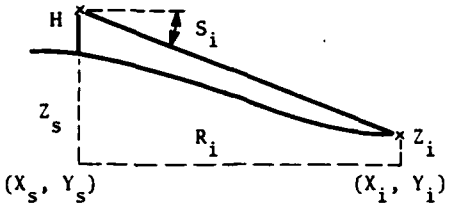
(c) The X-coordinate where the radial line intersects each row is given by $X_S + DX_1$ for ROW_1, and increments by DX_i for each subsequent row. If the intersection point is outside the viewing window, then skip to the next radial line, unless this is not the first intersection point and the previous point was visible: in that case first draw a line from the plot position to the previous point. Otherwise, the height Z_i at the intersection point is found by linear interpolation between the DTE points on either side, as shown.



(d) Similarly the horizontal range from the viewpoint to each intersection point is given by DR_1 for ROW_1, and increments by DR_i for each subsequent row.

(e) Now the slope S_i from the viewpoint to each intersection point, as shown below, is given by:

$$S_i = \frac{Z_i - (Z_S + H)}{R_i}$$



For ROW_N, the program finds the slope from the viewpoint to the point at the range limit, R; the height at the range limit, Z_R, is found by bilinear interpolation, then:

$$S_N = \frac{Z_R - (Z_S + H)}{R}$$

(f) Finally, the visible sections of the radial line are determined and plotted as follows: for each intersection point from 1 to N, if the slope S_i exceeds the maximum slope so far, then the point is visible from the viewpoint, so:

- set the maximum slope equal to S_i,
- if the previous intersection point was not visible, move the plot position to the current intersection point, and
- set a flag to indicate to the next step that the previous point <u>was</u> visible.

If the slope S_i is less than the maximum slope so far, then the point is <u>not</u> visible from the viewpoint, so

- if the previous point <u>was</u> visible, draw a line from the plot position to the previous point,
- set the flag to indicate to the next step that the previous point was <u>not</u> visible.

When all N points have been processed, if the last point was visible, then a line is drawn from the plot position to the last point.

The program repeats this procedure for the next radial line.

For radial lines which cross columns more often, the processing is similar, except that the values calculated in (b) above are in this case, as shown below, given by:

For COL_1, $DX_1 = REAL(COL_1 - 1) * GRDSPC - X_S$,

$DY_1 = DX_1 / TANAZ$,

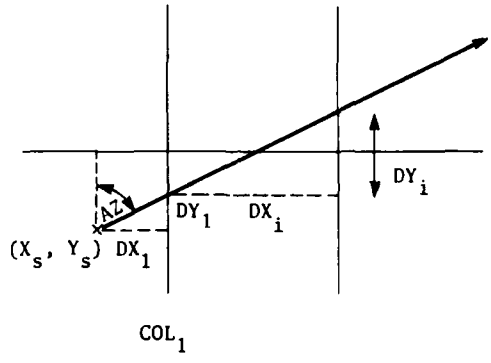
$DR_1 = SQRT(DX_1^2 + DY_1^2)$,

and for intermediate columns,

$DX_i = GRDSPC * CSTEP$,

$DY_i = DX_i / TANAZ$,

$DR_i = SQRT(DX_i^2 + DY_i^2)$.



The visibility diagram is drawn in "reverse-complement" mode, which means that the colour is the inverse of what previously was on the screen. Thus the diagram is mainly white (the inverse of the background color) except where the radial lines of the visibility diagram cross a contour line. The use of reverse complement mode means that the visibility diagram can be erased, simply by drawing the whole diagram again. A flag, DRAWFN, ensures that, if no visibility diagram has been drawn, the "erase visibility diagram" option has no effect.

4.3 Oblique view

In order to quickly generate an oblique view from any azimuth viewing angle, it is necessary (as will be shown below) to have two datasets containing the same DTE data; one stored by rows and the other stored by columns. A program (TRANSPOSE) has been provided in the secondary menu to generate one dataset from the other, to verify that the two datasets are consistent, or to list either dataset.

The oblique view program displays all the data for the selected area (N_R rows and N_C columns); the display is scaled such that:

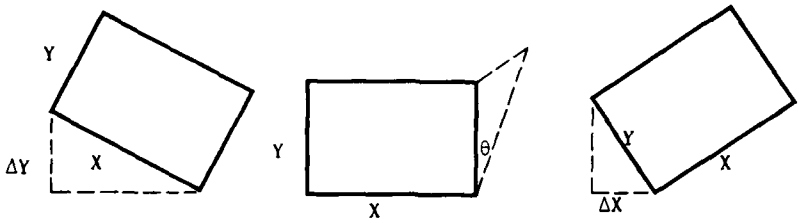
width = the width of the screen, or

height = the height of the screen.

The display is then centred on the screen, without knowledge of the terrain height. To correctly centre the display on the screen would require two passes through the program: the first to determine the vertical limits of

the display, and the second to scale and draw the display - this has not been considered necessary, as the display rarely reaches the limits of the screen.

Firstly, the user enters a terrain data sampling factor (N), an azimuth viewing angle (ϕ), an elevation viewing angle (θ), and a vertical exaggeration factor (V). The shape of the display on the screen depends on the azimuth viewing angle, as shown by the following examples:



ϕ = -30° ϕ = 0° ϕ = 30°

4.3.1 Scale

The program centres the display on the screen by scaling it to fit, either vertically or horizontally, and then setting the coordinates, on the screen, of the bottom left corner of the display (X_{ref}, Y_{ref}).

(a) Let X be the length of the bottom side of the display

(may be (N_R-1)*GRDSPC or (N_C-1)*GRDSPC depending on the azimuth viewing angle), and

(b) Let Y be the length of the left side of the display.

Define:

$$XC = \frac{\text{total width of display (m)}}{\text{GRDSPC}}$$

$$= \frac{X\cos|\phi| + Y\sin|\phi|}{\text{GRDSPC}}$$

and,

$$YC = \frac{\text{total height of display (m)}}{\text{GRDSPC}}$$

$$= \frac{(X\sin|\phi| + Y\cos|\phi|)\sin\theta}{\text{GRDSPC}}$$

(c) Let S be the spacing (cm) between each row or column on the screen. Initially set S so that the display fills the width of the screen:

$$S = \frac{\text{width of screen (cm)}}{XC}$$

Then if the height of the display (YC*S) is less than or equal to the height of the screen, centre the display vertically by setting:

$$X_{ref} = \Delta X * S,$$

$$Y_{ref} = \frac{\text{height of screen} - YC * S}{2} + \Delta Y * S$$

where, as shown in the above figure,

$$\Delta X = 0. \quad \text{if} \quad \phi \leq 0$$

$$= Y\sin|\phi| \quad \text{if} \quad \phi > 0, \text{ and}$$

$$\Delta Y = 0. \quad \text{if} \quad \phi \geq 0$$

$$= X\sin|\phi| \sin\theta \quad \text{if} \quad \phi < 0$$

(d) If however, the height of the display would exceed the height of the screen, reset S so that the display fills the height of the screen:

$$S = \frac{\text{height of screen (cm)}}{YC}$$

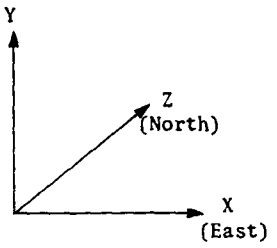
and centre the display horizontally by setting,

$$X_{ref} = \frac{\text{width of screen} - XC * S}{2} + \Delta X * S$$

$$Y_{ref} = \Delta Y * S$$

4.3.2 Transform

Reference 3 incorporates the following procedure for mapping the three-dimensional coordinates of a point of the DTE data into pixel position on the screen. The coordinate system used is:



At the intersection of row r and column c, let the coordinates be (X_D, Y_D, Z_D), where:

$$X_D = (c-1) * GRDSPC,$$

$$Z_D = (r-1) * GRDSPC,$$

and Y_D is the terrain elevation (m). X_D and Z_D are in metres relative to the SW corner of the DTE data.

The spacing S between each row or column was calculated earlier such that the display fits on the screen. It is used to convert the above coordinates from m to cm, as follows:

$$X_{rc} = X_D * (S/GRDSPC) = (c-1) * S$$

$$Y_{rc} = Y_D * V * (S/GRDSPC)$$

$$Z_{rc} = Z_D * (S/GRDSPC) = (r-1) * S$$

At the selected azimuth angle (ϕ) and elevation angle (θ) the coordinates (XP, YP) of the above point on the screen (in pixels) are given by:

$$XP = NINT((X_{rc}\cos\phi - Z_{rc}\sin\phi + X_{ref})*P) + 1$$

$$YP = NINT((X_{rc}\sin\theta\sin\phi + Z_{rc}\sin\theta\cos\phi + Y_{ref} + Y_{rc}\cos\theta)*P) + 1$$

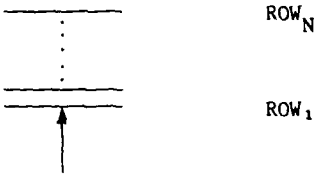
where

NINT means the nearest integer to the enclosed real number, and

P is the resolution of the screen (pixels/cm).

4.3.3 Mask

The oblique view algorithm works by drawing the first line of data (row or column) seen from the viewing azimuth angle, and then for successive lines of data, drawing any portions which are visible above the previous lines.



Viewing Azimuth = 0°

The first line of data is given by:

Azimuth Range	Lines	First Line
$\phi < 45°$ or $\phi \geq 315°$	rows	1
$45° \leq \phi < 135°$	columns	1
$135° \leq \phi < 225°$	rows	N_R
$225° \leq \phi < 315°$	columns	N_C

The program maintains a mask array with an entry for each pixel position across the screen; initially the array is set to zero. For each successive line, from the first, the program converts as above each point along the line (X_D, Y_D, Z_D) into a pixel position on the screen.

For each successive line, if the first point on the line is visible ($YP_1 \geq MASK\ (XP_1)$), then

- move the plot position to this point,
- set a flag (VISIBL) true to indicate, next time, that the previous point was visible, and
- set $MASK(XP_1)=YP_1$.

Otherwise, if the first point on the line is <u>not</u> visible, simply set VISIBL false. Then for subsequent point along the line; there are two possibilities.

(a) $XP_i = XP_{i-1}$

If YP_i is visible ($YP_i \geq MASK(XP_i)$) then

- if YP_{i-1} was visible, draw a line from YP_{i-1} to YP_i;
  if YP_{i-1} was not visible, draw a line from $MASK(XP_i)$ to YP_i,
- set VISIBL true,
- set $MASK(XP_i) = YP_i$.

If YP_i is not visible, simply set VISIBL false.

(b) $XP_i \neq XP_{i-1}$

The program checks for visibility and updates the mask (if necessary) at each pixel position from XP_{i-1} to XP_i (linearly interpolating from YP_{i-1} to YP_i), and, in addition, issues plot commands in the following two cases:

- if the previous point was not visible and the current point is visible, move the plot position to the current point,
- if the previous point was visible and the current point is not visible, draw a line to the previous point.

Finally, if YP_i *is visible, draw a line to* (XP_i, YP_i). Thus if the whole line from (XP_{i-1}, YP_{i-1}) to (XP_i, YP_i) is visible, a single plot command is issued to draw the line.

The above process is repeated for each point along the line, and for subsequent lines, until all the data has been processed.

5. DETAILED DESCRIPTION OF THE USER AND GRAPHICS INTERFACES

This section discusses the interfaces to the user and to the graphics device and consequently some special features of the RSX-11M/M-PLUS operating system on the PDP-11/34 used to develop the software.

All the programs in the suite of terrain modelling programs can be used with any one of several databases (DTE data). For each database, three datasets are required, as follows:

(a) Mapname.COL, containing the DTE data as 2 byte binary integers stored column by column (as supplied by ASR);

(b) Mapname.ROW, containing the same data but stored row by row (as generated by the TRANSPOSE option of the secondary menu);

(c) Mapname.TDD, containing

- the title of the map,
- the full name of the row by row dataset,
- the full name of the column by column dataset,

- the number of rows of DTE data,
- the number of columns of DTE data,
- the grid spacing of the DTE data (m).

Note that the ROW dataset is only actually required for the oblique view program, and then will only be used if the azimuth viewing angle is between 45° and 135°, or between 225° and 315°.

5.1 Menus

A menu system is provided to simplify the choice of database and program. The method of invoking the menu system depends on the particular installation.

5.1.1 Functional description

The primary menu (MAP) displays the databases available by locality name. A database is selected by typing the number preceding its locality name and then pressing the RETURN key. The secondary menu (PLOT) is immediately displayed, and lists the available terrain modelling programs which are selected in the same way. The current menu is terminated by typing the letter 'Q' and pressing the RETURN key.

When a terrain modelling program is stopped by the user, the PLOT menu is immediately redisplayed and another program choice can be made.

5.1.2 Implementation

The menu system is implemented as an RSX indirect command file which can be invoked under MCR or DCL. DCL must be able to be made the terminal's default command line interpreter. The menu text contains display control strings specific to the DEC VT200 (or VT100) family of terminals.

A terrain modelling program can be invoked in three ways (the contour plotting program and the Christmas Island database are used for the example):

```
        $ RUN CONTOUR/COMMAND='XMAS.TDD'
or
        $ RUN CONTOUR/COMMAND="CONTOUR XMAS.TDD"
or
        $ INSTALL CONTOUR       Task name is ...CON
        $ CONTOUR XMAS.TDD
```

In the last case if the file name were omitted then CONTOUR.DAT would be used as the database definition file name. The menu system uses the first method.

5.2 User terminal input

All the terrain modelling programs obtain user input via the TERMIN module which implements a common style for prompting and entering data. The module also overcomes a restriction of FORTRAN-77 numeric input of not being able to distinguish between a null response by the user (ie the user only presses the RETURN key in response to a prompt, which is the usual method for requesting a default value) or a zero entered by the user. Thus had this restriction not been overcome it would not have been possible to override the default value with a value of zero.

5.2.1 Functional description

There are four prompt and read formats:

(a) integer input with default,
```
DEFAULT:   50  ==>
```

(b) 8-digit coordinate with default,
```
DEFAULT: 35008700  ==>
```

(c) integer input without default,
```
==>
```

and

(d) user confirmation,
```
Press RETURN when ready.
```

A call to the input subprograms may optionally specify a maximum permissible value in which case the minimum permissible value is zero. The user is prompted for input until a permissible value is entered.

For example,

```
  DEFAULT:    0  ==> -90
Value out of range [0..359] -- try again
  DEFAULT:    0  ==>
```

Where the prompt displays a default value, pressing only the RETURN key requests the default value. If no default value is displayed (as in format (c)) then a permissible value must be explicitly entered. Format (d) signifies that no input value is required.

There are five subprograms for requesting terminal input. GETINT (get integer), GETPOS (get coordinate position) and GETOPT (get option) correspond to the first three formats above. GETREL accepts real arguments but calls GETINT to request input in integer form. WAITOK (wait for confirmation) uses format (d) and is used to stall program execution until the user confirms to continue.

The following are examples of calls to the TERMIN subprograms:

```
IARG = 0
CALL GETINT (IARG,10)          !default=0; range=0..10
CALL GETINT (IARG,0)           !default=IARG; no range check
ARG = 5.0
CALL GETREL (ARG,0.0)          !default=5.0; no range check
CALL GETOPT (N,7)              !nodefault; range=0..7
IX = 6700
IY = 9800
CALL GETPOS (IX,IY)            !default=6700°800; no range check
CALL WAITOK
```

5.2.2 Implementation

Before calling any TERMIN subprogram the object LTERM in the named common IO must be initialised with the LUN of the user's terminal. For example,

```
COMMON /IO/ LTERM
DATA LTERM/5/
```

Terminal input is read as a character string. A null response is signalled by a blank string. If not blank, the string is used as the source of a formatted read statement. The logging of conversion errors is suppressed by calling the PDP-11 FORTRAN-77 OTS routine ERRSET. If an error occurs the TERMIN module logs the error and prompts for more input.

5.3 Graphics line driver

The graphics processor acts as a "carriage control device" connected to the computer by a serial terminal line. Normally, if a string of data is sent to such a device by a program via a FORTRAN formatted WRITE statement, program execution is suspended until the I/O has completed. Since I/O is not a CPU-intensive operation, if no other programs are running on the computer the CPU is lying idle for most of the time required to transmit the data. The GDRIVER module was implemented to allow program execution to proceed concurrently with graphics I/O.

Every PDP-11 FORTRAN-77 WRITE statement results in a request to the operating system to perform I/O, ie no buffering of I/O data is done by the FORTRAN OTS. The GDRIVER module removes this unnecessary overhead by buffering up to 250 bytes of I/O data before requesting I/O to be performed.

5.3.1 Functional description

The GDRIVER module accepts strings of 8-bit ASCII data (via subprogram WRITEG) for transmission to a line device (eg graphics processor, plotter, etc.) such that the application program is free to continue processing without waiting for I/O completion. The data is not interpreted in any way and no extra characters are transmitted except in the following cases:

- Trailing blanks (SP) are stripped from strings passed to WRITEG,
- UPDATG ensures that all strings are transmitted, and additionally transmits a terminating carriage return (CR), and
- CLOSEG calls UPDATG before closing the channel.

(a) Subprogram OPENG is used to initialise the GDRIVER module and the transmission channel,

```
CALL OPENG (DEVICE,DEVLEN,LUN,EFN,LOG)
```

where DEVICE is a CHARACTER*80 string identifying the line device; DEVLEN is the length of the device string; LUN is the logical unit number to use for the line device; EFN is the event flag number to use for the line device; LOG is the logical unit number to use for logging errors. For example,

```
CALL OPENG ('TT2:',4,1,1,5)
```

Note that LUN should not be assigned or opened prior to calling OPENG but LOG should be already assigned to a terminal or an open sequential file. Since EFN can be set and cleared asynchronously with program execution, it should not be used by any other part of the program.

(b) Subprogram WRITEG is used to send a string to the line device,

```
CALL WRITEG (STRING,STRLEN)
```

where STRING is a CHARACTER*n string; STRLEN is the length of the string (<=n). For example,

```
      CHARACTER*20 STRING
      WRITE (STRING,10) IX,IY
   10 FORMAT ('M',I3,',',I3)
      CALL WRITEG (STRING,LEN(STRING))
      CALL WRITEG ('RC',2)
```

(c) Subprogram UPDATG is used to update the line device (ie to synchronise the application program with the line device),

```
CALL UPDATG
```

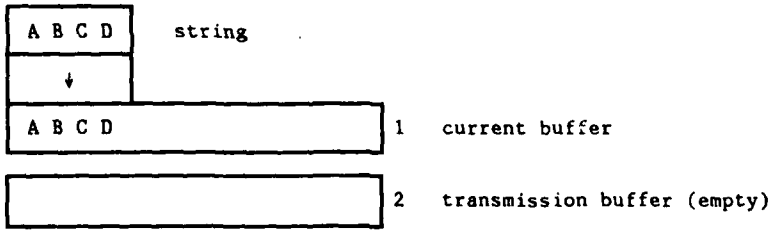
(d) Subprogram CLOSEG is used when the current application is complete,

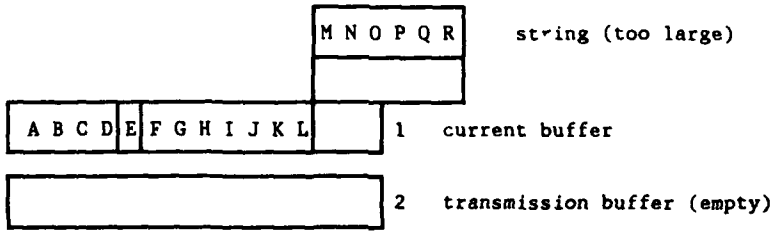
```
CALL CLOSEG
```

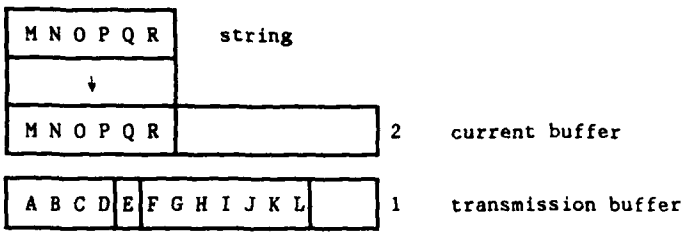
thus ensuring all data has been sent before the package is reinitialised or the program terminates.

5.3.2 Implementation

In general, PDP-11 RSX performs I/O directly between the task buffer containing the data and the hardware device controller, ie a copy of the data is not taken. Thus a dual buffering technique is required. The "current" buffer is used to buffer strings received via WRITEG. The "transmission" buffer is specified in the request to perform I/O (QIO). Rather than copy the current buffer to the transmission buffer the roles of the buffers are swapped when it is appropriate to so. A typical sequence of events in GDRIVER is shown below:







The device specified in the call to OPENG must be controlled by the full-duplex terminal driver. Data is transmitted to the device by calling the QIO executive directive directly. The main advantage in doing this is that the QIO directive returns control to the calling program when directive processing is complete but before I/O processing is complete. The write-passall function code is specified in the QIO directive which signals the full-duplex terminal driver not to perform any checks on the output data and not to wrap lines. This avoids spurious carriage control characters being inserted in the I/O data. By using the null (0) control character as the vertical format control character, no carriage control characters are added by the terminal driver at all. However, some commands for many graphics devices require termination by the transmission of the next command or, if there are no more commands, by a carriage return character. So UPDATG requests that the next transmission be terminated with a carriage return character by using the '+' (43) character as the vertical format control character.

5.4 Software interrupt

Since some graphics applications can take a long time to run, a method of interrupting the program was developed. Pressing CTRL-C on the user's terminal effects an interrupt in one of two ways. While the program is prompting the user for input, CTRL-C gains the attention of the command line interpreter (DCL or MCR) in the normal manner, allowing a CLI command to be entered but otherwise does not interrupt program execution. While the program is generating a graphics display, CTRL-C prints "*Cancel*" on the terminal and signals the program to cancel its graphics processing.

5.4.1 Functional description

The module CANCEL implements the asynchronous cancellation request by the user. Three subprograms and one named common are used to control CTRL-C processing. The boolean flag in the named common ASYNC is true if a CTRL-C has been trapped by the program. INICAN is a FORTRAN- callable subroutine to initialise the CANCEL module (CTRL-C is still trapped by CLI). ENCANC is a FORTRAN-callable subroutine to enable CTRL-C trapping. DSCANC is a FORTRAN-callable subroutine to disable CTRL-C trapping.

Declare named common in FORTRAN program as:

```
LOGICAL*1 CANCEL
COMMON /ASYNC/ CANCEL
```

and use:

```
IF (CANCEL) ...
```

The storage for CANCEL must be 1 or 2 bytes. In PDP-11 FORTRAN-77 the default storage is 2. The compiler switch /I4 causes 4 bytes of storage to be allocated and used in tests. If /I4 must be used, explicitly declare the type of CANCEL as LOGICAL*1 or LOGICAL*2.

Examples of the subroutine calls:

```
      INTEGER*2 LUN,EFN
      DATA LUN/5/,EFN/1/

      CALL INICAN (LUN,EFN)
      ...
      CALL ENCANC
      DO 100 ...
         IF (CANCEL) GOTO 200
      ...
  100 CONTINUE
  200 CALL DSCANC
```

5.4.2 Implementation

The boolean flag CANCEL once tested and found to have the value .TRUE. should be set to .FALSE. by assignment, by calling ENCANC, or by calling DSCANC, before testing for another cancel request.

The terminal is assumed to be already attached on entry to each routine and is immediately detached to help ensure the subsequent attach request succeeds. Errors are not reported to the calling program.

The event flag supplied to INICAN is used asynchronously by the Asynchronous System Trap (AST) service routine to signal the receipt of CTRL-C on the user's terminal. This event flag must not be used by other parts of the program.

The save (SAV) attribute is given to all program sections (.PSECTs) which contain asynchronously used code or data. This instructs the task builder to place the code and data in the root segment so that it is always available in an overlaid program.

FORTRAN tasks are, by default, detached from the user's terminal, except during task I/O. INICAN attaches the terminal allowing unsolicited characters to be placed in the terminal's type-ahead buffer.

The subroutine ENCANC attaches the user's terminal, specifying an unsolicited-input-character AST such that an unsolicited character, other than CTRL-C, is placed in the terminal's type-ahead buffer (or input buffer if a read is in progress). On receipt of CTRL-C, the terminal's type-ahead buffer is flushed, and the AST routine is entered without terminating a read request if one is in progress.

5.5 Program termination

All the terrain modelling programs use the subprograms OKSTOP and ERSTOP to terminate execution. These subprograms return exit status to the operating system or parent task such that the reason for termination can be determined. Two reason codes are used: normal termination (SUCCES) and termination due to fatal error (SEVERE).

6. POSSIBLE ENHANCEMENTS

A number of possible enhancements to the displays have been identified, some could be made to the existing software, if required, some would require additional data to be provided, and one would require a particular type of display device.

6.1 Line diagrams

The reader may have noticed that the four displays provided are all line diagrams. Shading or colour filled polygons have not been used for several reasons:

(a) Time

Filling polygons with colour or shading is normally done by the graphics device but nevertheless takes much more time than drawing straight line segments.

(b) Information

The finished product looks good, but conveys no more information than a line diagram, particularly if the colour of the lines were to be varied, as described in the next Section.

(c) Simplicity

The algorithms for line drawings are more simple, and hence require less CPU time.

(d) Flexibility

The graphics device need not have a hardware polygon fill capability.

6.2 Colour options

6.2.1 Oblique and perspective view

If required, the height of each displayed line segment could be used to set the colour of the line segment (the line segment may need to be split into two segments of different colours if the line spans a height boundary).

Also if vegetation or "going" data were stored, for each map, then the user could select to set the colour of the lines according to

- height
- vegetation, or
- going.

Another option which might be useful for the oblique view, would be to draw 1 km lines (for example) in colour, with a label at one end. This would enable the user to read the coordinates of features or areas of interest.

6.2.2 Contour plot

If vegetation information is important and a method can be developed for making vegetation data available (a particularly suitable method would

be to have another dataset which contains a vegetation type, and a percentage coverage, for each data square) then the vegetation data could be displayed by filling each data square with colour, according to the vegetation type, and then drawing the contour lines The vegetation colours would be displayed first, because filling squares with colour without erasing contour lines would be very time-consuming.

Alternatively, if "going" is important, then the colour of each data square could be set according to the "going" within that square.

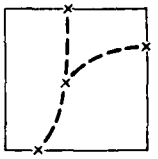
6.3 Additional information

If it is considered vital for additional information such as

- spot heights,
- roads, railways, rivers, cliffs - any type of linear feature

to be added to one or more of the displays, then the following method is recommended.

It would be desirable for efficient processing if such data were accessible by data square, otherwise the program must search through all the data and display only those which lie within the selected area. Also, the majority of data squares will have no railways, for example, but if a railway is present at least two coordinate pairs will be needed to define the railway within the square, eg in the example shown:



four coordinate pairs are needed.

The recommended approach is to have:

(a) A spot height array (N_R-1 by N_C-1) to store a spot height (if present), or zero otherwise, and

(b) A linear feature array (N_R-1 by N_C-1) which is an array of pointers, or zero if there are no linear features within the grid square. The pointer would store the address where the following information is stored:

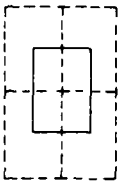
- the number of different linear features within the square, and for each different feature,
- the type, and
- sufficient pairs of coordinates to enable plotting of the feature within the square.

6.4 Overlap of adjoining areas

No attempt has been made in the present software to handle the case where part of a visibility diagram is on one map, and the other part on an adjoining map. The problem is non-trivial because the lines of longitude

are not parallel. The approach normally adopted is to store an additional "map" which includes half of the first map and half of the second, the user selects the data which fully encloses the required area. Thus the terrain data is duplicated twice

- stored by rows and by columns, and
- with overlap, as shown.



The problem of longitude lines not being parallel should not be critical in that, for this application, exact accuracy is not necessary. Therefore it should be possible to have a terrain data input routine which, when requested for a row or column of data (of specified length) opens another dataset (if required) and reads the rest of the row or column from the new dataset. The terrain dataset names would have to be an alphanumeric code, to enable the program to simply identify the required adjoining data. This approach would be preferable, because data storage requirements are large already, (particularly if vegetation, going, spot heights and linear features need to be stored), without another duplication of the DTE data.

6.5 Panning (of contour plot or perspective view)

The contour plot could pan in any direction, and the perspective view could pan left or right, if the display device used a "bit-mapped" architecture, as explained below:

Because the contour plot is generated one data square at a time, it would be easy to step along an additional row or column, if the origin of the display could then be shifted accordingly. Thus it would be necessary for the display device to have a "bit-mapped" architecture, ie the display is stored in the device's internal memory and mapped onto the screen; shifting the origin changes the picture which appears on the screen. Thus if the new row or column information is placed in the correct position, and the origin is shifted, then the contour plot could pan, in any direction, one row or column at a time.

The cursor control keys could be used to control the panning, or cross-hair controls on the graphics device.

Similarly, the perspective view could be shifted one azimuth step at a time.

7. CONCLUSION

The software described in this memorandum provides a simple, efficient means of utilising present-day computers and graphics terminals for the display of Digital Terrain Elevation data. In particular, there is the flexibility to

- locate a viewpoint for a visibility diagram anywhere on the map,
- generate an oblique view from any azimuth or elevation angle,

- generate a perspective view from a viewpoint (located anywhere on the map), looking in any direction, with a field-of-view up to 180°.

- check the terrain profile between any two points on the map.

Simple algorithms have been utilised to enable the displays to be drawn as quickly as possible, while still providing significant enhancements over previously available software. Also, some indication has been given of further possible improvements for the displays.

REFERENCES

No.	Author	Title
1	-	"Digital Terrain Modelling Reference Manual". Army Survey Regiment, June 1985
2	-	"Algorithm 483 Masked Three-Dimensional Plot Program with Rotations". Communications of the ACM, Volume 17, No 9, September 1974
3	Dowling, M.J.	"An Efficient Contour Plotting Subroutine". ERL-0150-TM, July 1980

THIS IS A BLANK PAGE

APPENDIX I

CRITIQUE OF THE ASR SOFTWARE

This Appendix presents an outline of some of the algorithms used by the ASR software, together with reasons why the software is not suitable for the current application.

I.1 Contour plotting

The program sets up contour plotting squares, and within each square it carries out the following operations:

(a) the maximum and minimum height of the four corner points is calculated, and hence the contour levels which lie within the square,

(b) linear interpolation is used to find the points at which the contour intersects each of the four sides of the square, in the order left, bottom, right, top;

(c) if there are two such points, a straight line is drawn from one to the other, which is correct, however:

(d) if there are three such points, a straight line is drawn from point 1 to point 2, which could be quite wrong (see figure 13), and

(e) if there are four such points, straight lines are drawn from point 1 to point 2 and from point 3 to point 4, which again could be quite wrong (figure 13).

Also, the grid which is constructed for contour plotting is completely independent of the terrain data, which is very inefficient because then a bilinear interpolation procedure has to be used to calculate the height at each corner of each contour plotting square.

I.2 Visibility diagram

The user nominates a range limit and sector, and the program subdivides the sector into 50 azimuth angles, and then steps through 50 steps along each radial line; if the latest point is visible, then a line is drawn from the previous point to the latest point. Thus if the terrain is visible from the viewpoint out to the range limit, a straight line will be formed by drawing 50 small straight lines.

A number of other problem areas were identified:

(a) The azimuth increment is fixed at 7.2° for a 360° sector. A better approach is to let the user select the azimuth increment, which again gives the user the option of a quick look, or a detailed diagram,

(b) Where possible the small straight line segments should be concatenated into a longer straight line segment, to reduce the number of plot commands sent to the display device, because the speed of the display may be limited by transmission speed on the line from the computer to the display device,

(c) It requires an enormous amount of code to select the particular subset of data which is required to draw each fan segment. For a 5 km maximum range, and a full circle visibility diagram, the data required is a 101 x 101 INTEGER*2 array, which is acceptable. If a larger maximum range is required, then the storage requirements could be

reduced by drawing the left side and then the right side of the visibility diagram.

(d) The maximum range of the diagram is always divided into 50 increments, which means that diagrams with a small maximum range are not drawn any faster.

I.3 Oblique view

Reference 1 states that "The software does not permit view rotation, but rather, it permits generation of a plot based on one of the four viewing edges. Do not enter azimuths as multiples of 45.0".

Also, when the program was run with the Christmas Island data, a number of errors occurred:

(a) When 50 profiles were requested with an aspect angle of 30° and a viewing azimuth of 0°, the program stopped partway through the plot as a result of array subscript errors.

(b) When 100 profiles were requested with an aspect angle of 30° and a viewing azimuth of 180°, the program stopped because of a "record number outside range".

(c) On several occasions, the hidden line technique failed near the end of the plot.

As a result, no attempt was made to understand the algorithm used by this program. A new program was written based upon reference 2.

I.4 Perspective view

Reference 1 states that the perspective view program "is the least stable program". Again, it requires an enormous amount of code to select the particular subset of data which is needed to draw the required view.

The algorithm selected for drawing an oblique view in the new software package could be straightforwardly extended to draw a perspective view; consequently no attempt was made to understand the algorithm used by the original program.

I.5 General

(a) If a new set of data is to be selectable by the ASR programs, a number of changes must be made within subroutine CARTO, and each program must be recompiled. The process should be much simpler.

(b) The ASR programs are designed to use terrain data which includes three slopes in addition to the height at each point; this leads to much higher storage requirements, greater processing requirements, and is simply unnecessary for the current application, because sufficient accuracy can be obtained using only height data at 100 m grid spacing.

APPENDIX II

MODULE DEPENDENCIES

II.1 Contour plotting program

```
CONTOUR     CLRSCR
            CONLBL
            CONTUR      DSCANC
                        ENCANC
                        ERSTOP
                        GETINT
                        PLTCCD      PLOT
                        SETCLR      COLOUR
                        UPDATE
            ENDPLT
            FAN         ERSTOP
                        GETREL
                        HEIGHT
                        PLOT
                        REPMOD
                        UPDATE
            GETDB
            GETINT
            GETOPT
            GRID        PLOT
                        REPMOD
                        UPDATE
            INICAN
            INIPLT
            IWID
            LOCATE
            MAPNAM
            OKSTOP
            SYSFOV      GETINT
                        GETREL
```

II.2 Oblique view program

```
OBLIQUE     CLRSCR
            DSCANC
            ENCANC
            ENDPLT
            GETDB
            GETINT
            GETOPT
            GETREL
            INICAN
            INIPLT
            IWID
            OBLLBL
            OKSTOP
            PLOT3D      PLTPIX
            UPDATE
```

II.3 Perspective view program

PERSPEC	CLRSCR	
	ENDPLT	
	FOV	GETREL
	GETDB	
	GETOPT	
	HAG	IWID
	INICAN	
	INIPLT	
	LOCATE	
	MAPDAT	HEIGHT
	OKSTOP	
	PVIEW	COLOUR
		DSCANC
		ENCANC
		HEIGHT
		PERLBL
		PLTPIX
		UPDATE
	RNGLBL	

II.4 Profile program

PROFILE	ABSTEP	
	CLRSCR	
	ENDPLT	
	GETDB	
	GETINT	
	GETREL	
	INIPLT	
	LOCATE	
	MAPNAM	
	OKSTOP	
	PROFDR	COLOUR
		PLOT
	PROLBL	
	UPDATE	

II.5 Utility subroutines

GETDB	INIDB	GETMCR (system routine)
	OPENDB	

GETINT)		
GETREL)		ERRSET (system routine)
GETPOS)	GETXXX	IWID
GETOPT)		OKSTOP

LOCATE	ABS10	
	GETPOS	
	REL1	

APPENDIX III

FILE CONTENTS

File	Subroutine	Entry Points
*CONTOUR.FTN	SYSFOV	
	GRID	
	CONTUR	
	PLTCCD	
	SETCLR	
	FAN	
*OBLIQUE.FTN	PLOT3D	
*PERSPEC.FTN	FOV	HAG
	MAPDAT	
	PVIEW	
*PROFILE.FTN	PROFDR	
	ABSTEP	
RSXUTE.FTN	GDRIVER	OPENG, CLOSEG, WRITEG, BREAKG
	STOP	OKSTOP, ERSTOP
UTILITY.FTN	GETDB	GETDB, INIDB, OPENDB
	TERMIN	GETINT, GETREL, GETPOS, GETOPT, GETXXX
	LOCATE	
	ABS1O	
	REL1	
	HEIGHT	
	IWID	
VECTRIX.FTN	PLTPIX	
	PLOT	
	INIPLT	
	ENDPLT	
	CLRSCR	
	UPDATE	
	COLOUR	
	TEXT	
	REPMOD	
	CONLBL	
	MAPNAM	
	OBLLBL	
	PERLBL	RNGLBL
	PROLBL	
CANCEL.MAC	CANCEL	INICAN,DSCANC,ENCANC

* Main program precedes the subroutines.

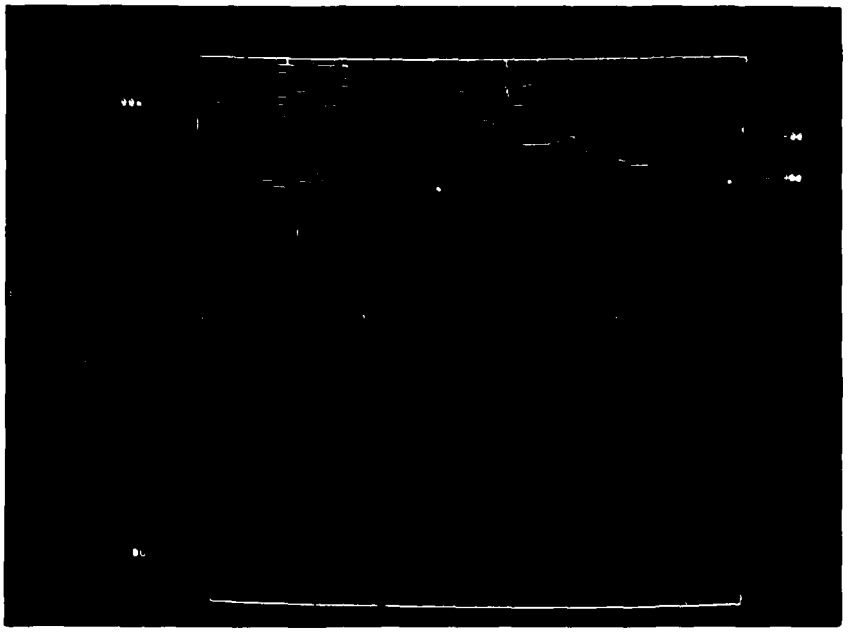


Figure 1. Contour Plot: Normal map scale, 20 m contours drawn using every point of the terrain data (1 min 50 s)

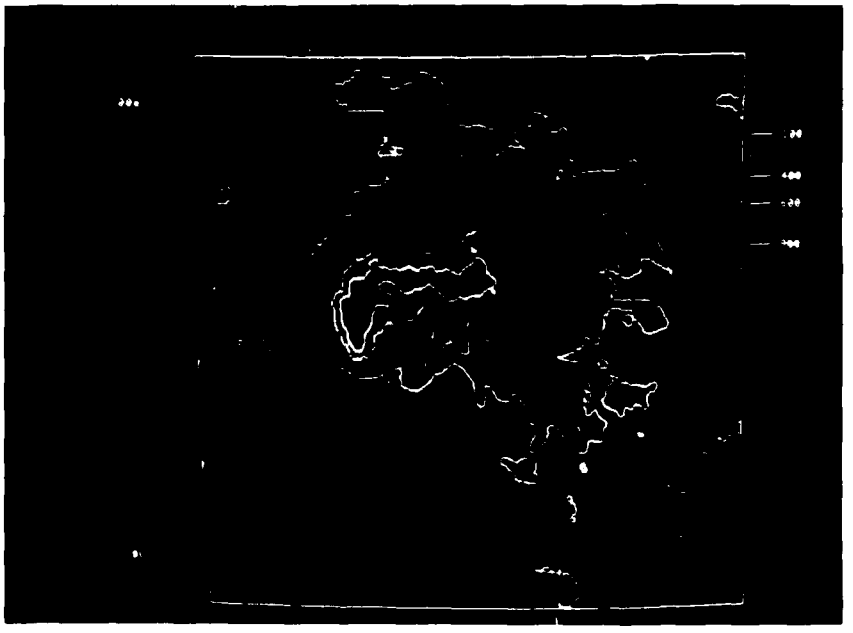


Figure 2. Contour Plot: Half map scale, 20 m contours drawn using every point of the terrain data (7 min 10 s)

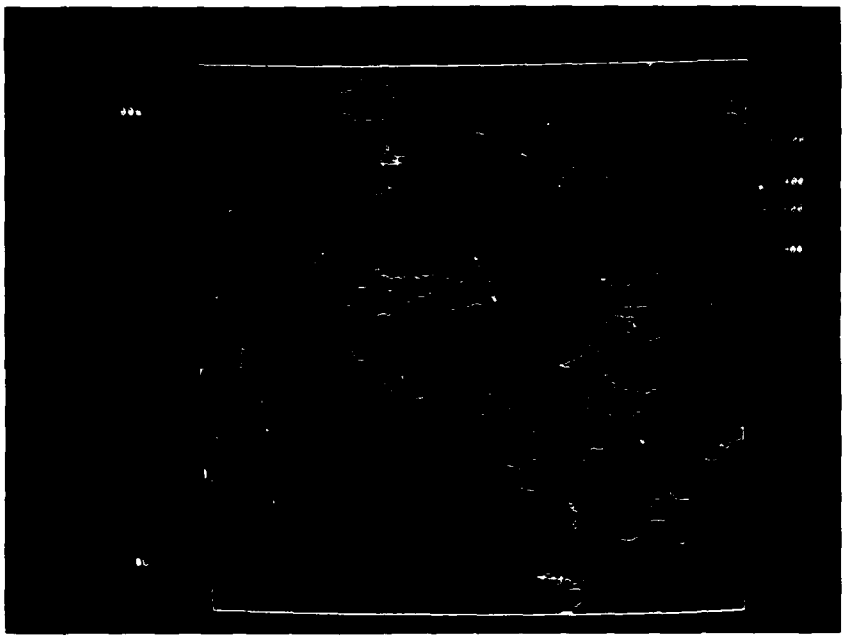


Figure 3. Contour Plot: Hlaf map scale, 50 m contours drawn using every point of the terrain data (3 min 10 s)

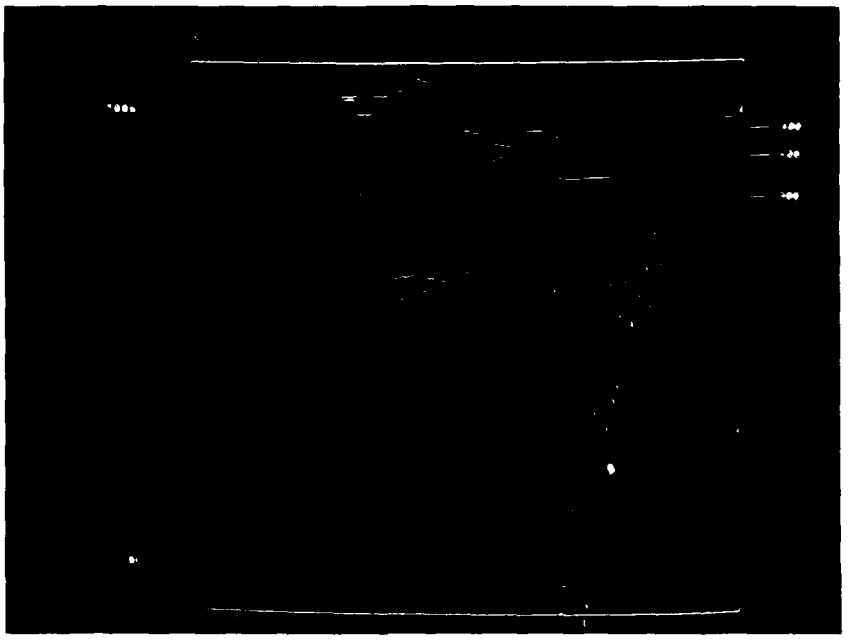


Figure 4. Contour Plot: Half map scale, 50 m contours drawn using every fifth point of the terrain data (30 s)

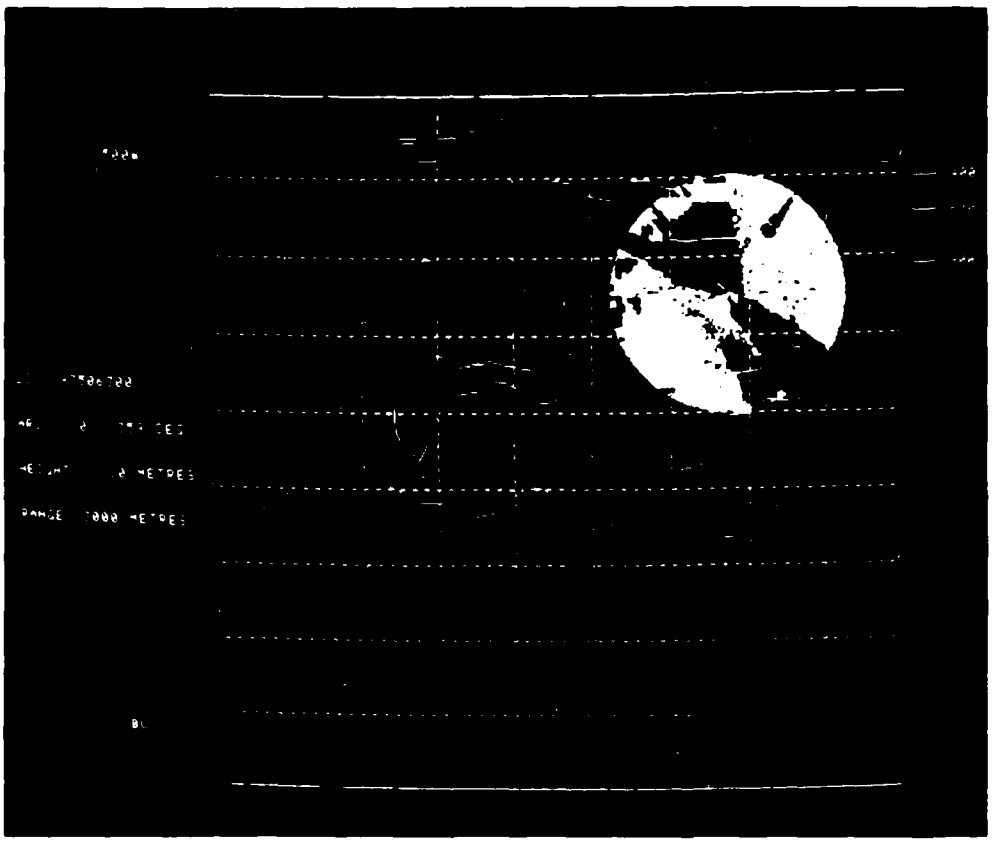


Figure 5. *Contour Plot: Half map scale, 20 m contours drawn using every fifth point of the terrain data (66 s). Also, a visibility diagram drawn with a 1° increment (18 s), and a grid (1 s)*

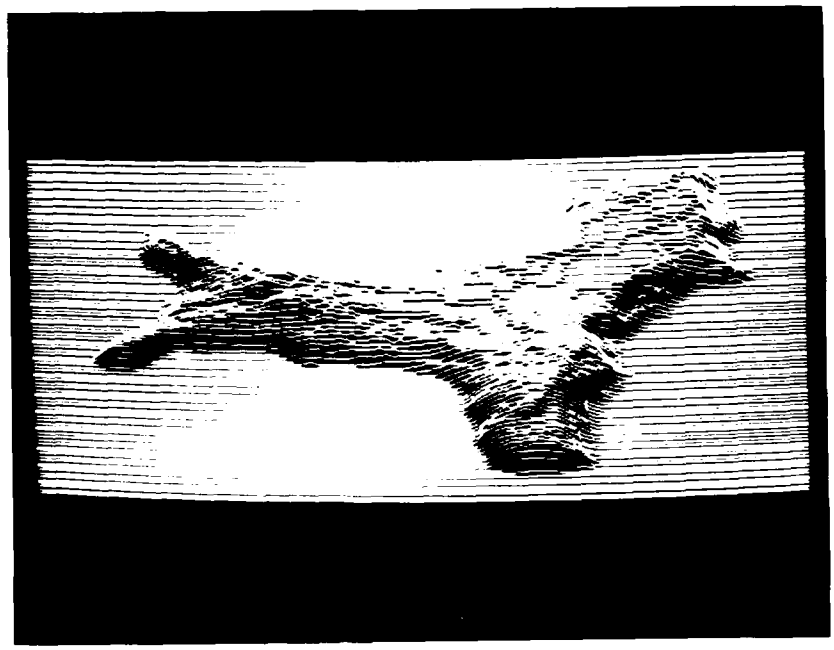


Figure 6. Oblique view from the South at 30° elevation, using every second point of the terrain data (1 min 40 s)

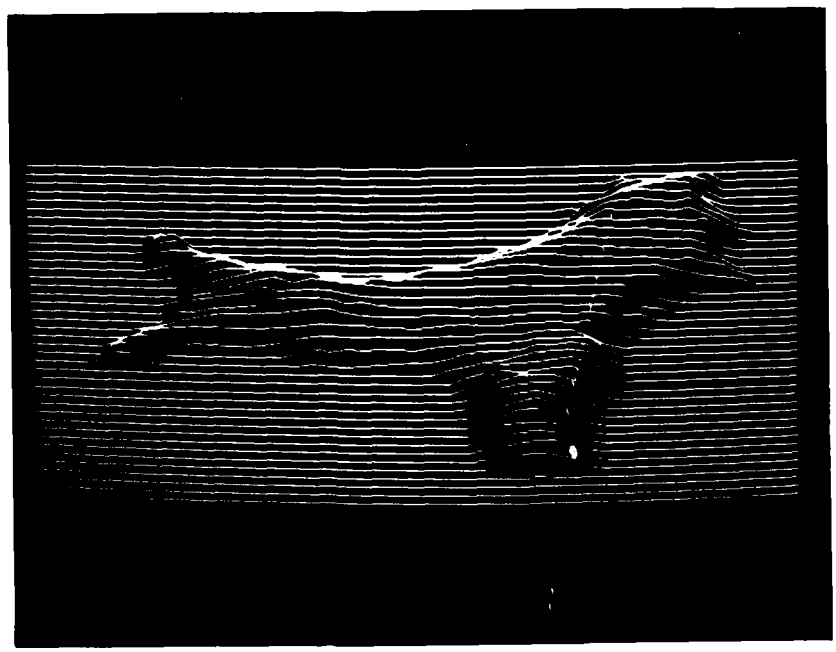


Figure 7. Oblique view from the South at 30° elevation, using every fifth point of the terrain data (20 s)

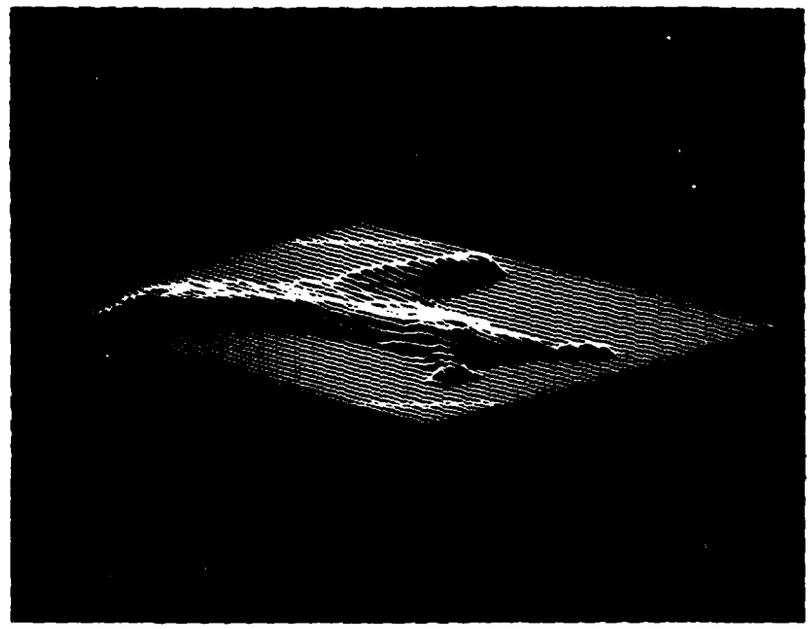


Figure 8. Oblique view from the Northwest at 15° elevation, using every fifth point of the terrain data (20 s)

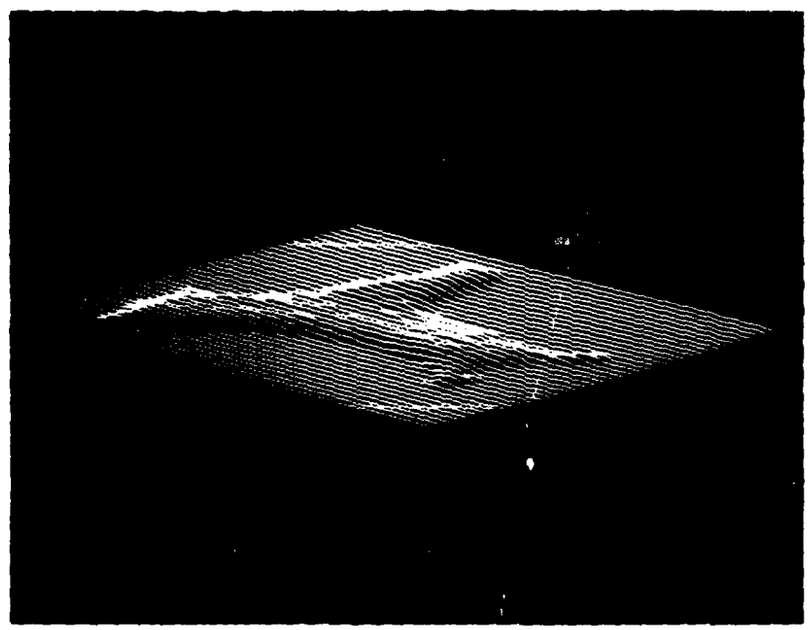


Figure 9. Oblique view from the Northwest at 15° elevation, using every fifth point of the terrain data, with no vertical exaggeration (20 s)

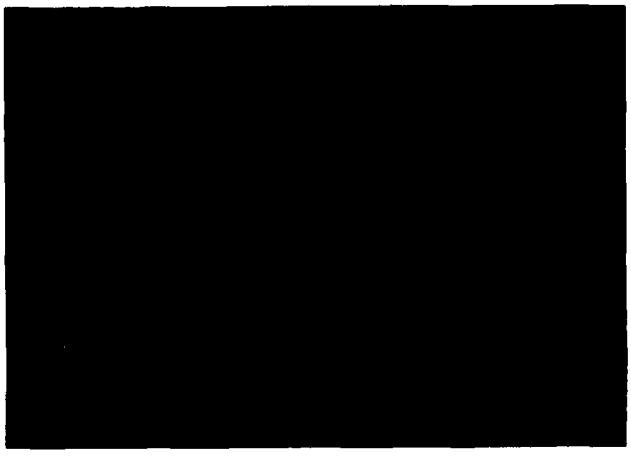


Figure 10. Perspective view (55 s)

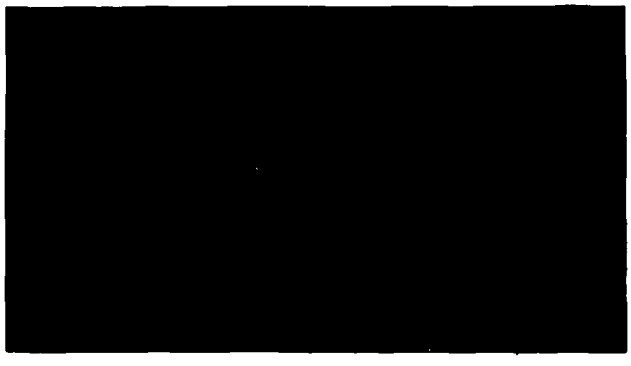


Figure 11. Profile (5 s)

Figure 12. Perspective view (constant range increments)

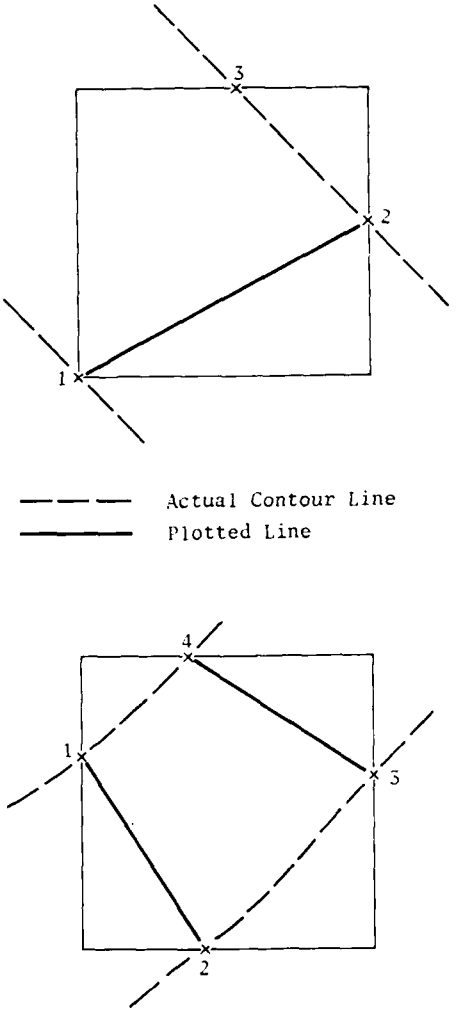


Figure 13. ASR Contour Plotting Algorithm

DISTRIBUTION

	Copy No.
DEPARTMENT OF DEFENCE	
Defence Science and Technology Organisation	
Chief Defence Scientist	1
Deputy Chief Defence Scientist	
Controller, External Relations, Projects and Analytical Studies	
Superintendent, Science Programs and Administration	
Counsellor, Defence Science, London	Cnt Sht Only
Counsellor, Defence Science, Washington	Cnt Sht Only
Electronics Research Laboratory	
Director, Electronics Research Laboratory	2
Superintendent, Radar Division	3
Superintendent, Electronic Warfare Division	4
Principal Officer, Computer Research Group	5
Principal Officer, Systems Studies Group	6
Weapons Systems Research Laboratory	
Mr P.F. Calder, Combat Systems Integration Group	7
Army Office	
Scientific Adviser - Army	8
Director, Operational Analysis - Army	9
Director, Operations - Army	10
Director, Operational Requirements - Army	11
Director, Communications - Army	12
Director, Electronic Procurement - Army	13
Director, Command and Control Systems - Army	14
Director, Survey - Army	15

Director General, Army Development (NSO), Russell Offices
for ABCA Standardisation Officers

UK ABCA representative, Canberra 16

US ABCA representative, Canberra 17

Canada ABCA representative, Canberra 18

NZ ABCA representative, Canberra 19

CO, Army Survey Regiment, Bendigo 20

Commander, 1 DIV, Enoggera 21

Commander, Engineering Development Establishment 22

Joint Intelligence Organisation (DSTI) 23

Libraries and Information Services

Librarian, Technical Reports Centre, Defence Central
Library, Campbell Park 24

Document Exchange Centre
Defence Information Services Branch for:

Microfiche copying 25

United Kingdom, Defence Research Information Centre 26 - 27

United States, Defense Technical Information Center 28 - 39

Canada, Director, Scientific Information Services 40

New Zealand, Ministry of Defence 41

National Library of Australia 42

Main Library, Defence Research Centre Salisbury 43 - 44

Librarian, DSD 45

UNITED KINGDOM

British Library, Lending Division 46

Authors 47 - 48

Spares 49 - 53

DOCUMENT CONTROL DATA SHEET

Security classification of this page: UNCLASSIFIED

1 DOCUMENT NUMBERS

AR Number: AR-004-868

Series Number: ERL-0393-TM

Other Numbers:

2 SECURITY CLASSIFICATION

a. Complete Document: Unclassified

b. Title in Isolation: Unclassified

c. Summary in Isolation: Unclassified

3 TITLE

FAST, COLOUR DISPLAYS OF DIGITAL TERRAIN ELEVATION DATA

4 PERSONAL AUTHOR(S):

M.J. Dowling and
S.C.C. McAuley

5 DOCUMENT DATE:

November 1986

6 6.1 TOTAL NUMBER OF PAGES 46

6.2 NUMBER OF REFERENCES: 3

7 7.1 CORPORATE AUTHOR(S):

Electronics Research Laboratory

7.2 DOCUMENT SERIES AND NUMBER

Electronics Research Laboratory
0393-TM

8 REFERENCE NUMBERS

a. Task: ARM 83/109

b. Sponsoring Agency: 1159/82

9 COST CODE:

684529/133

10 IMPRINT (Publishing organisation)

Defence Research Centre Salisbury

11 COMPUTER PROGRAM(S) (Title(s) and language(s))

12 RELEASE LIMITATIONS (of the document):

Approved for Public Release

Security classification of this page: UNCLASSIFIED

Security classification of this page: UNCLASSIFIED

13 ANNOUNCEMENT LIMITATIONS (of the information on these pages):

No limitation

14 DESCRIPTORS:		15 COSATI CODES:
a. EJC Thesaurus Terms	Colour, Datum (elevation), Terrain, Mapping, Australia.	08060
b. Non-Thesaurus Terms	Digital Terrain Elevation (DTE)	

16 SUMMARY OR ABSTRACT:
(if this is security classified, the announcement of this report will be similarly classified)

Software has been developed to provide the following displays of digital terrain elevation data with a given map area:

(a) A contour plot with one or more visibility diagrams,

(b) An oblique view of the map area from any azimuth or elevation angle,

(c) A perspective view of a specified sector from a selected viewpoint, and

(d) A point-to-point line-of-sight profile.

Emphasis has been placed on making the software easy to use, on generating the displays as quickly as possible, and on making effective use of colour.

Security classification of this page: UNCLASSIFIED


END
DATE
FILMED
12 87